Apple II
Technical Notes

Developer Technical Support

HyperCard IIGS
#1: Corrections to the *Script Language Guide*

Written by: Dan Strnad & Matt Deatherage March 1991

This Technical Note corrects the *HyperCard IIGS Script Language Guide* from Addison-Wesley.

Appendix A: External Commands and Functions

Page 317: ReturnStat

Developers who worked with the beta version of HyperCard IIGS on Volume V of the Developer CD (or volume 4 of Developer Essentials) should pay special attention to the use of the `returnStat` parameter documented on page 317 of the manual, as this method for using HyperCard's error-reporting facilities was not present in beta versions of HyperCard.

Page 318: HyperCard IIGS callbacks

Before describing the callbacks, the *Script Language Guide* says that the first parameter to each callback is the parameter block pointer that HyperCard IIGS passes to the XCMD or XFCN. This is not correct; the XCMD/XFCN parameter block is not passed to callback routines. Each callback uses only the parameters supplied with its description.

Pages 318–324: Callback descriptions

The numbers listed for each callback are actually decimal numbers, not hexadecimal. There should not be a "$" in front of each number.

Pages 325–330: Beep, an example XCMD

Although there are "beep" sample XCMDs provided with the *HyperCard IIGS Script Language Guide*, they do not necessarily build and execute unmodified. Specifically, depending on your compiler, there could be a linking problem with the Pascal and C XCMDs as given in the manual.

XCMDs and XFCNs are code resources, and are therefore subject to the limitations listed in Apple IIGS Technical Note #86, Risking Resourceful Code. The specific problem here is that most Pascal and C compilers will create at least three segments: `~globals`, `~arrays`, and `main`. An XCMD or XFCN can only have one segment and the entry point must come first.

Not only must you link all the object segments into one segment, but you must specifically extract the entry point and link it **first**. HyperCard will pass control to the first byte of the loaded XCMD or XFCN, and therefore this must be the entry point. The samples in Appendix A point this out in the code.

Actual buildable sample source for the "beep" XCMDs is available in APW and MPW IIGS format on Volume VI or later of the Developer CD Series (or volume 5 or later of Developer Essentials). A complete APW C sample is included below.

An APW Sample XCMD: "CBeep"

CBeep.c

```
/*----------------------------------------------------------------------

  file CBeep.c

  This XCMD has the following syntax:

    CBeep       beep once
    CBeep ##    beep n times
    CBeep ?     display usage information
    CBeep !     display version information

  Copyright Apple Computer, Inc.  1989-1991
  All Rights Reserved.

----------------------------------------------------------------------*/

#include <types.h>
#include <MiscTool.h>
#include <GSOS.h>
#include <HyperXCMD.h>

/*
    Globals
*/

int _toolErr;
XCMDPtr gParamPtr;


/*
    Forwards
*/
pascal void CBeep();



/*  We place the entry point function in its own segment, so the linker can
    extract it and ensure that it's first in the load file. */

segment "EntrySeg"

/*
   This is the entry point to the program.  Make sure this procedure
   comes first in the final OMF resource because this is where HyperTalk
   will be jumping in.

   For a really simple XCMD you could just put the code all in here, but
```

```
   for cleanliness' sake this example calls another routine from here.


*/
pascal void EntryPoint(paramPtr)
XCMDPtr paramPtr;
{
  CBeep(paramPtr);
}

/*  All other code & data is placed in the "Main" segment   */

segment "Main"



/*  The actual CBeep function.  Interpret parameters and beep the speaker   */

pascal void CBeep(paramPtr)
XCMDPtr paramPtr;
{
  short     beepCount;
  short     counter;
  Str255    str;

  char  *formStr    = "\pAnswer \"FORM: CBeep {count}\"";
   char   *versionStr =  "\pAnswer  \"CBeep  XCMD  v1.0\"  &  return  &  \"(c)  1991  Apple
Computer, Inc.\"";

  gParamPtr = paramPtr;     /* put in a global for easy access in other funcs   */

  if (paramPtr->paramCount > 0) {
    ZeroToPas(*(paramPtr->params[0]), &str);

    beepCount = 0;

    if (str.text[0] == '?')         /* test for special characters  */
      SendCardMessage(formStr);
    else if (str.text[0] == '!')
      SendCardMessage(versionStr);

    else beepCount = StrToNum(&str);        /* not a special - take as # of beeps */
  }
  else beepCount = 1;   /* no count, assume one */

  beepCount = (beepCount <= 15) ? beepCount : 15;   /* limit 15 beeps   */

  for (counter = 0; counter < beepCount; counter++) SysBeep();
}
```

CBeep.r

```
/*****************************************************************/
/*
/* CBeep.r
/*
/* Copyright (C) 1991
/* Apple Computer, Inc.
/* All Rights Reserved
/*
/* Rez source for building XCMDs.
/*
/*****************************************************************/

#include "types.rez"
```

```
read $801E (1, convert) "CBeep.omf";

resource rResName ($0001801E) {
            1,
            { 1, "CBeep";
            }
};
```

Make file

```
* ---------------------------------------------------------------
*
*  This makefile will build C XCMDs for HyperTalk
*
*  Copyright Apple Computer, Inc.  1991
*  All Rights Reserved.
*
*  Builds:  CBeep
*  This makefile depends on a .r file called CBeep.r to act
*  as a source for the resource compiler.

compile +t +e CBeep.c keep=CBeep

* ---------------------------------------------------------------
*
*  The compilers will output 3 or more segments:  main, containing code;
*  and ~globals and ~arrays containing data.  This line ensures that
*  everything gets put back into the main segment.
*
*  In addition, it specifically links the EntryPoint procedure FIRST,
*  ahead of any globals or data structures.

* The linker line is very long - make sure you use all of it

linkiigs -x -lseg main CBeep.root(@EntrySeg) CBeep.root(@Main) CBeep.root(@~arrays)
CBeep.root(@~globals) 2/CLib -lib 2/CLib -o CBeep.omf

compile CBeep.r keep=CBeep.rsrc

* now use your favorite resource utility to copy the XCMD from CBeep.rsrc
* into your stack
```

Further Reference

- *HyperCard IIGS Script Language Guide*
- Apple IIGS Technical Note #86, Risking Resourceful Code
- HyperCard IIGS Technical Note #2, Known HyperCard Bugs